Insight:

Variables & Variable Scope

Knowing difference between different Ruby variables and especially their scopes is essential to becoming efficient metaprogrammer.

Variable types in Ruby:

• global variables	starts with “$”
• class variables	starts with “@@”
• instance variables	starts with “@”
• local variables	no prefix
• constants	starts with uppercase letter

Global variables

```ruby
$global_var = "Yes I am!"

class MyClass
    puts "Accessible in class def? #{$global_var}"

    def foo
        puts "Accessible in method? #{$global_var}"
    end

    def self.foo
        puts "Accessible in class method? #{$global_var}"
    end

    $another_global_var = "Defined in MyClass."
end

                                                                           # Accessible in class def? Yes I am!
MyClass.new.foo                                                            # Accessible in method? Yes I am!
MyClass.foo                                                                # Accessible in class method? Yes I am!

puts $another_global_var                                                   # "Defined in MyClass."
```

Global variable scope

- Global variables have global scope!
- Once initialised, they will be available from anywhere in your code.
- They will not be undefined until your code stops running.

Global variables

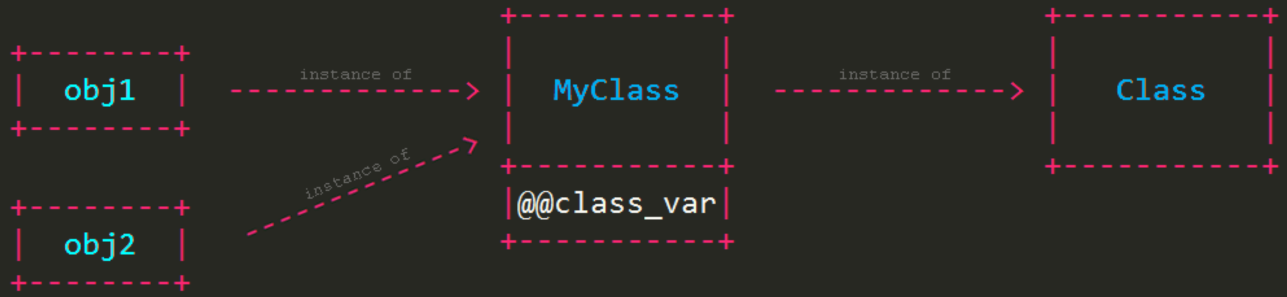
```
puts global_variables                              # [:$;, :$-F, :$@, :$!, :$SAFE, :$~, :$&, :$`, :$', :$+, :
```

Variable types in Ruby:

• global variables	starts with “$”
• **class variables**	starts with “@@”
• instance variables	starts with “@”
• local variables	no prefix
• constants	starts with uppercase letter

Class variables

```ruby
class MyClass
    @@class_var = "Shared by all instances of MyClass class."

    def class_var
        @@class_var
    end

    def self.class_var
        @@class_var
    end

    def big_bang
        @@class_var = "Changed by instance."
    end
end


obj1, obj2 = MyClass.new, MyClass.new

obj1.class_var                                                # "Shared by all instances of MyClass class."
obj2.class_var                                                # "Shared by all instances of MyClass class."
MyClass.class_var                                             # "Shared by all instances of MyClass class."

obj1.big_bang

obj1.class_var                                                # "Changed by instance."
obj2.class_var                                                # "Changed by instance."
MyClass.class_var                                             # "Changed by instance."
```



Class variable scope

- All instances of the class share class variables.
- Class variables are accessible anywhere in the body of class definition, including any type of method definitions.
- Will be in the scope, when “self" is referring to your class, instance of the your class or descendant of your class’s class or instance.

Class variables

```ruby
class MyClass
    @@class_var = "MyClass class var"

    class InnerClass
        def class_var
            @@class_var
        end
    end

    def class_var
        @@class_var
    end
end

MyClass.new.class_var                       # "MyClass class var"
MyClass::InnerClass.new.class_var           # NameError: uninitialized class variable


module M
    def class_var
        @@class_var
    end
end

class MyClass
    include M
    @@class_var = "MyClass class var"
end

MyClass.new.class_var                       # NameError: uninitialized class variable
```

```ruby
class Parent
    @@class_var = "Class var in Parent."
end

class Child < Parent
    def class_var
        @@class_var
    end
end

Child.new.class_var                         # "Class var in Parent."


class MyClass
    @@class_var = "Class variable."
end

MyClass.class_variables                     # [:@@class_var]
```

Variable types in Ruby:

• global variables	starts with "$"
• class variables	starts with "@@"
• **instance variables**	starts with "@"
• local variables	no prefix
• constants	starts with uppercase letter

Instance variables

```
class MyClass
    attr_accessor :instance_var

    def initialize
        @instance_variable = "Instance variable"
    end

    def self.instance_var
        @instance_variable
    end

    def big_bang
        @instance_variable = "Changed value!"
    end
end


obj1, obj2 = MyClass.new, MyClass.new

obj1.instance_var                                             # "Instance variable"
obj2.instance_var                                             # "Instance variable"

obj1.big_bang

obj1.instance_var                                             # "Changed value!"
obj2.instance_var                                             # "Instance variable"

MyClass.instance_var                                          # nil
```

Instance variable scope

- Only available when "self" is referring to:
  - the instance of the class or;
  - an instance of descendant class;
  - nowhere else.
- They live as long as your class instances live.

Variable types in Ruby:

• global variables	starts with "$"
• class variables	starts with "@@"
• instance variables	starts with "@"
• **local variables**	no prefix
• constants	starts with uppercase letter

Local variable scope

- Defined for the current block of code. Such as: method definition, lambda, class declaration, etc.
- When execution exists the scope of the block, local variables are removed.
- Only live for as long as the current block is being executed.

Local variables

```ruby
local_var = "Local variable"

def some_method(arg)
    # local_var                             - not accessible
    arg                                     # local variables are created for method arguments
    some_var = 123
end

# args, some_var                            - not available, don't exist anymore


block = lambda do
    block_var = "in lambda def"
    puts local_var
end

puts block_var                              # NameError: undefined local variable or method `block_var' for main:Object
block.call                                  # "Local variable"



class MyClass
    local_var = "in class MyClass"
end

MyClass.class_eval { puts local_var }           # NameError: undefined local variable or method `local_var' for MyClass:Class
```

Variable types in Ruby:

global variables	starts with “$”
class variables	starts with “@@”
instance variables	starts with “@”
local variables	no prefix
constants	starts with uppercase letter

Constants

```ruby
TOP_LEVEL = "Top level."

class MyClass
    IN_CLASS = "In class definition body."

    def foo
        ANOTHER_ONE = "Another one in class."
    end
end

module MyModule
    ANOTHER_ONE = "Another one in module."
end
```

```
+- main
|  +- TOP_LEVEL
|  |
|  +- MyClass
|  |   +- IN_CLASS
|  |   +- ANOTHER_ONE
|  |
|  +- MyModule
|      +- ANOTHER_ONE
```

Constants scope

- They are removed when their “parents” are removed.
- Constants defined in top level scope will be available as far as the program is running.